# aerofit-project-b-rekha-raghu

May 26, 2025

# 1 Business Case: Aerofit - Data Exploration and Visualisation

## 1.1 Business Problem:

Perform descriptive analytics to create a customer profile for each AeroFit treadmill product by developing appropriate tables and charts. For each AeroFit treadmill product, construct two-way contingency tables and compute all conditional and marginal probabilities along with their insights/impact on the business.

## 1.2 Importing Libraries:

[13]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
```

## 1.3 Loading Dataset:

[14]:
```
df=pd.read_csv('aerofit')
df
```

[14]:
```
    Product  Age  Gender  Education MaritalStatus  Usage  Fitness  Income  \
0     KP281   18    Male         14        Single      3        4   29562
1     KP281   19    Male         15        Single      2        3   31836
2     KP281   19  Female         14     Partnered      4        3   30699
3     KP281   19    Male         12        Single      3        3   32973
4     KP281   20    Male         13     Partnered      4        2   35247
..      …   …      …          …           …      …        …       …
175   KP781   40    Male         21        Single      6        5   83416
176   KP781   42    Male         18        Single      5        4   89641
177   KP781   45    Male         16        Single      5        5   90886
178   KP781   47    Male         18     Partnered      4        5  104581
179   KP781   48    Male         18     Partnered      4        5   95508

     Miles
0      112
1       75
2       66
```

```
3        85
4        47
..       …
175     200
176     200
177     160
178     120
179     180

[180 rows x 9 columns]
```

### 1.4 Dataset Shape:

[4]:
```python
print("\nDataset Shape (rows, columns):", df.shape)
```

```
Dataset Shape (rows, columns): (180, 9)
```

### 1.5 Datatype of Each Column:

[5]:
```python
print("Column Data Types:\n", df.dtypes)
```

```
Column Data Types:
 Product          object
Age              int64
Gender          object
Education        int64
MaritalStatus   object
Usage            int64
Fitness          int64
Income           int64
Miles            int64
dtype: object
```

### 1.6 Structure of Dataset:

[6]:
```python
print(df.info())
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 180 entries, 0 to 179
Data columns (total 9 columns):
 #   Column         Non-Null Count  Dtype
---  ------         --------------  -----
 0   Product        180 non-null    object
 1   Age            180 non-null    int64
 2   Gender         180 non-null    object
 3   Education      180 non-null    int64
```

```
 4   MaritalStatus  180 non-null    object
 5   Usage          180 non-null    int64
 6   Fitness        180 non-null    int64
 7   Income         180 non-null    int64
 8   Miles          180 non-null    int64
dtypes: int64(6), object(3)
memory usage: 12.8+ KB
None
```

### 1.7 Checking Missing values:

[7]:
```
print("\nMissing Values Count:\n", df.isnull().sum())
#--->We can coclude that there are Zero null values
```

```
Missing Values Count:
 Product          0
Age              0
Gender           0
Education        0
MaritalStatus    0
Usage            0
Fitness          0
Income           0
Miles            0
dtype: int64
```

[8]:
```
print(df.duplicated().sum())
#---> No duplicate Rows found in the given DataFrame
```

```
0
```

### 1.8 Relationship between features and product purchased:

[38]:
```
Product_vs_Gender=df.groupby(["Gender","Product"]).size().unstack().T
Product_vs_Gender
```

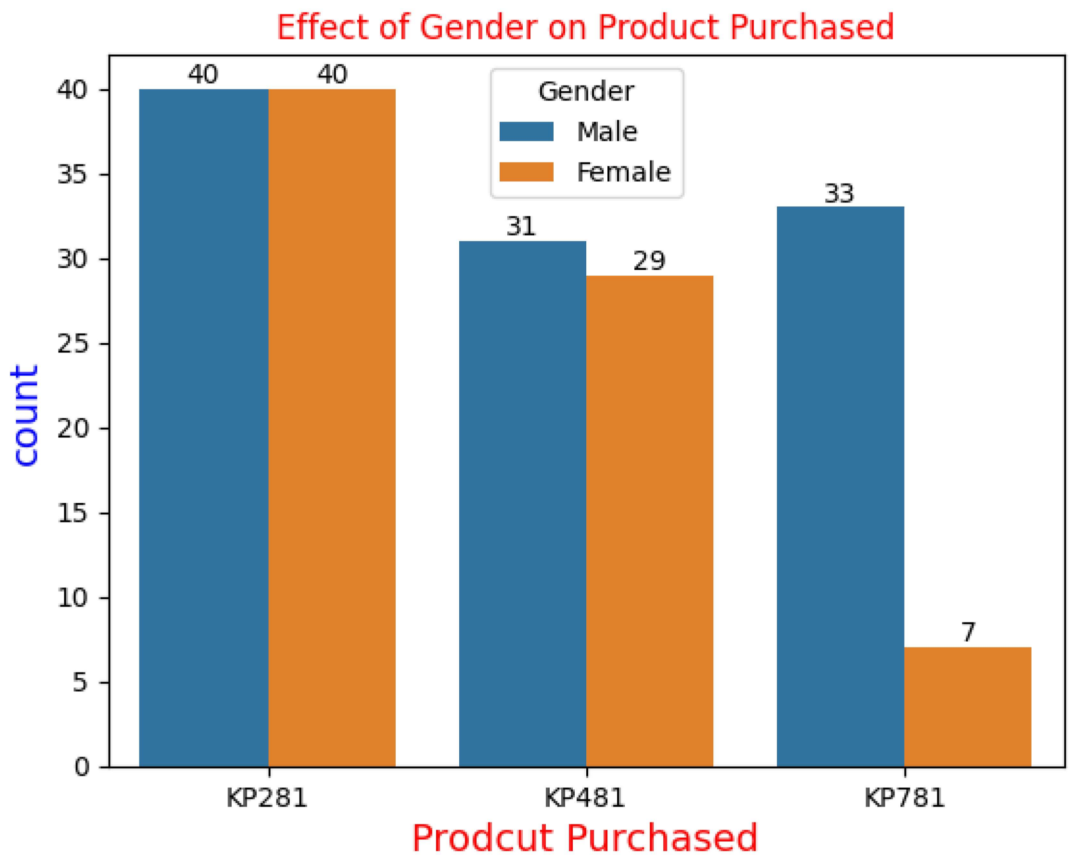
[38]:

| Gender | Female | Male |
|---|---|---|
| Product | | |
| KP281 | 40 | 40 |
| KP481 | 29 | 31 |
| KP781 | 7 | 33 |

KP281 can be marketed to a broad audience,KP481 should maintain its balanced approach,KP781 could be marketed toward male customers

[39]:
```
#Visual Representation
ax=sns.countplot(x='Product',hue='Gender',data=df)
```

```
for bar in ax.containers:
  ax.bar_label(bar,fmt='%d')
plt.title("Effect of Gender on Product Purchased",color="red")
plt.xlabel("Prodcut Purchased", color="red", fontsize=14)
plt.ylabel("count", color="blue", fontsize=14)
plt.show()
```



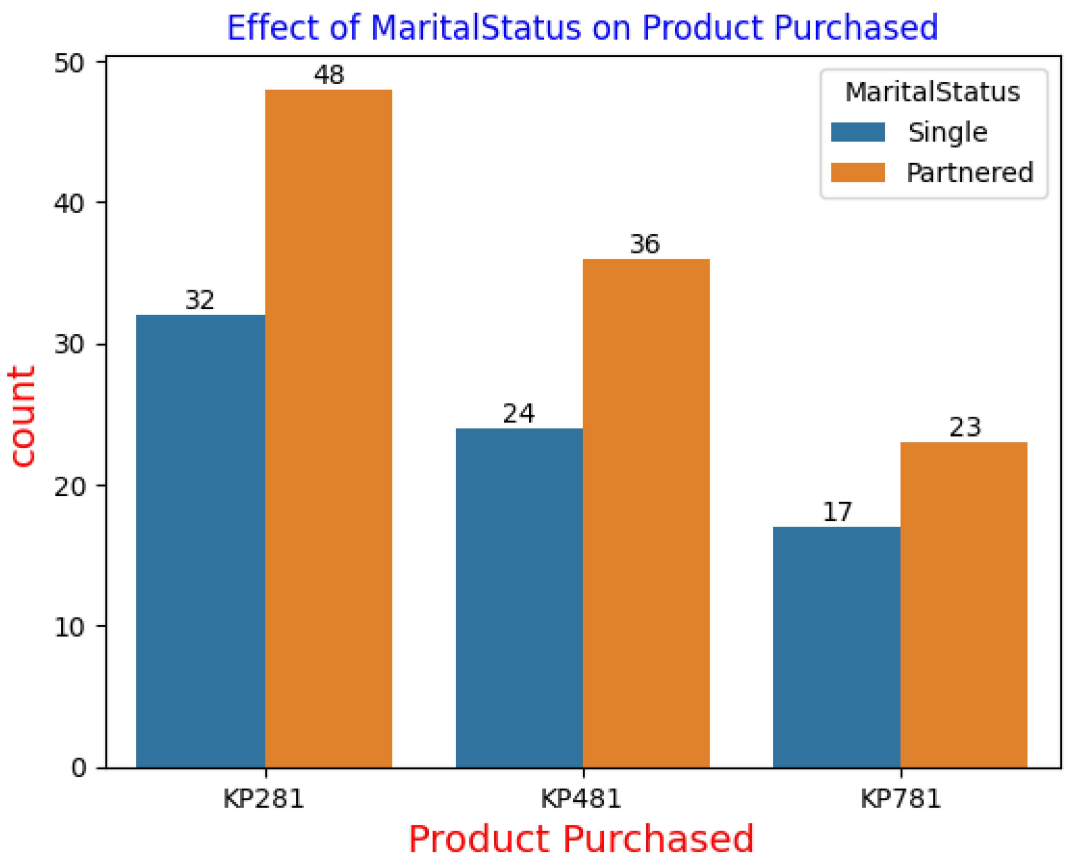

[40]:
```
Product_vs_MaritalStatus=df.groupby(["MaritalStatus","Product"]).size().
  ↪unstack().T
Product_vs_MaritalStatus
```

[40]:

| MaritalStatus | Partnered | Single |
|---|---|---|
| Product | | |
| KP281 | 48 | 32 |
| KP481 | 36 | 24 |
| KP781 | 23 | 17 |

KP281 and KP481 should be marketed toward households,KP781's marketing should highlight as marital status seems less influential in its purchase.

[41]:
```
ax=sns.countplot(x='Product',hue='MaritalStatus',data=df)
for bar in ax.containers:
  ax.bar_label(bar,fmt="%d")
plt.title("Effect of MaritalStatus on Product Purchased",color="blue")
plt.xlabel("Product Purchased", color="red", fontsize=14)
plt.ylabel("count", color="red", fontsize=14)
plt.show()
```




[19]:
```
Product_vs_Age=df.groupby(["Age","Product"]).size().unstack()
Product_vs_Age.head()
```

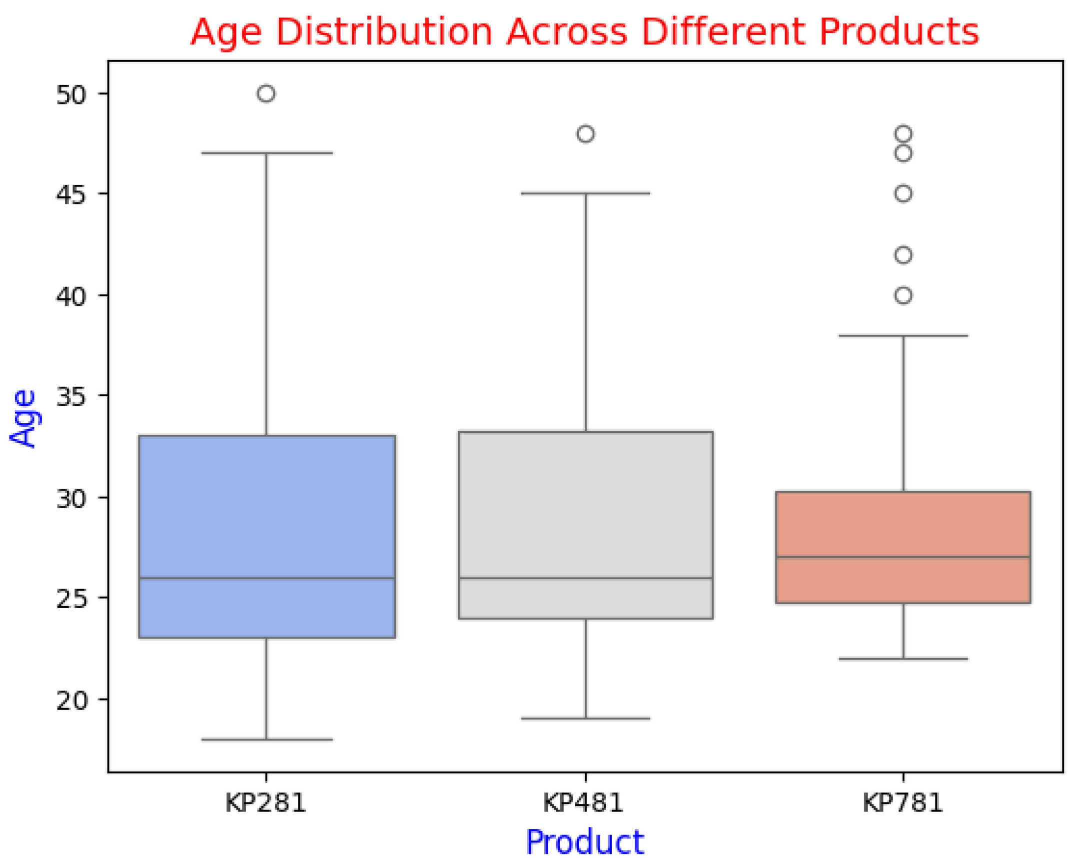
[19]:

| Product | KP281 | KP481 | KP781 |
|---|---|---|---|
| Age | | | |
| 18 | 1.0 | NaN | NaN |
| 19 | 3.0 | 1.0 | NaN |
| 20 | 2.0 | 3.0 | NaN |
| 21 | 4.0 | 3.0 | NaN |
| 22 | 4.0 | NaN | 3.0 |

KP281 should be marketed toward younger first-time buyers, emphasizing affordability & ease of use. KP481 can target young adults (19–21) looking for a more balanced fitness option. KP781 might be best promoted as a high-performance treadmill for experienced users aged 22+, highlighting advanced features.

[43]:
```
sns.boxplot(x='Product', y='Age', data=df, palette='coolwarm')
plt.title("Age Distribution Across Different Products", fontsize=14,␣
 ↪color='red')
plt.xlabel("Product", fontsize=12, color='blue')
plt.ylabel("Age", fontsize=12, color='blue')
plt.show()
```

```
<ipython-input-43-9527d5435632>:1: FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be removed in
v0.14.0. Assign the `x` variable to `hue` and set `legend=False` for the same
effect.

  sns.boxplot(x='Product', y='Age', data=df, palette='coolwarm')
```



[42]:
```
Grouped_data=pd.
↪concat([Product_vs_Gender,Product_vs_MaritalStatus],keys=["Gender","MaritalStatus"])
Grouped_data
```

[42]:

| | | Female | Male | Partnered | Single |
|---|---|---|---|---|---|
| | Product | | | | |
| Gender | KP281 | 40.0 | 40.0 | NaN | NaN |
| | KP481 | 29.0 | 31.0 | NaN | NaN |
| | KP781 | 7.0 | 33.0 | NaN | NaN |
| MaritalStatus | KP281 | NaN | NaN | 48.0 | 32.0 |
| | KP481 | NaN | NaN | 36.0 | 24.0 |
| | KP781 | NaN | NaN | 23.0 | 17.0 |

### 1.9 Selecting Continuous Variables:

[44]:
```
continuous_vars=df.select_dtypes(include=['int64','float64']).columns
continuous_vars
```

[44]:
```
Index(['Age', 'Education', 'Usage', 'Fitness', 'Income', 'Miles'],
dtype='object')
```

### 1.10 Compute Correlation Between Continuous Variables and Product:

[50]:
```
#Converting categorical Product into numerical
df['Product_Code']=df['Product'].astype('category').cat.codes
correlation_matrix=df[['Product_Code','Age','Education','Usage','Fitness','Income','Miles']].
↪corr()
correlation_matrix['Product_Code']
```

[50]:
```
Product_Code    1.000000
Age             0.032225
Education       0.495018
Usage           0.537447
Fitness         0.594883
Income          0.624168
Miles           0.571596
Name: Product_Code, dtype: float64
```

### 1.11 Strongest Correlations:

Product_Code & Income (0.6241) → Higher income influences treadmill model choice.

Product_Code & Fitness (0.5949) → People with better fitness levels tend to prefer specific treadmill models.

Product_Code & Miles (0.5716) → Customers who cover more miles tend to favor certain products.

Product_Code & Usage (0.5374) → Frequency of treadmill usage affects product choice.

## 1.12 Moderate Correlation:

Product_Code & Education (0.4950) → Higher education level slightly influences treadmill model preference.

## 1.13 Weakest Correlation:

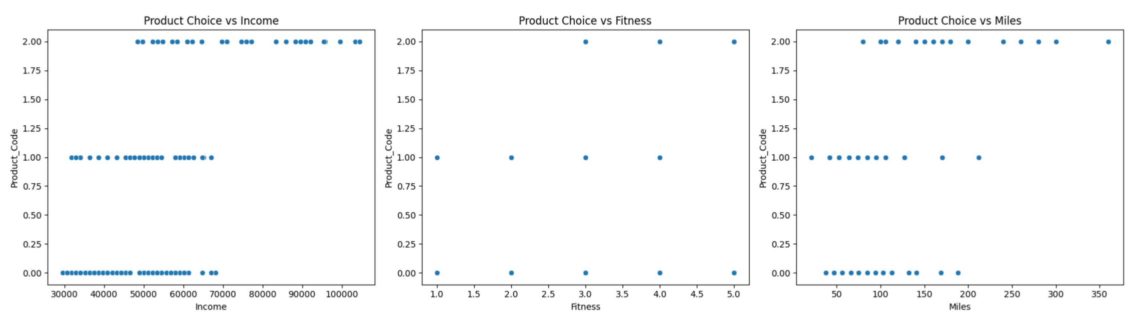
Product_Code & Age (0.0322) → Age has almost no impact on treadmill model preference.

Higher-income customers tend to prefer premium treadmill models. Lower-income buyers mostly choose entry-level treadmills, while mid-range earners show mixed preferences.

Visual representation of continuous variables and output:

---

[65]:
```
fig, axes = plt.subplots(1, 3, figsize=(18, 5))

sns.scatterplot(x="Income", y="Product_Code", data=df, ax=axes[0])
axes[0].set_title("Product Choice vs Income")

sns.scatterplot(x="Fitness", y="Product_Code", data=df, ax=axes[1])
axes[1].set_title("Product Choice vs Fitness")

sns.scatterplot(x="Miles", y="Product_Code", data=df, ax=axes[2])
axes[2].set_title("Product Choice vs Miles")

plt.tight_layout()
plt.show()
```



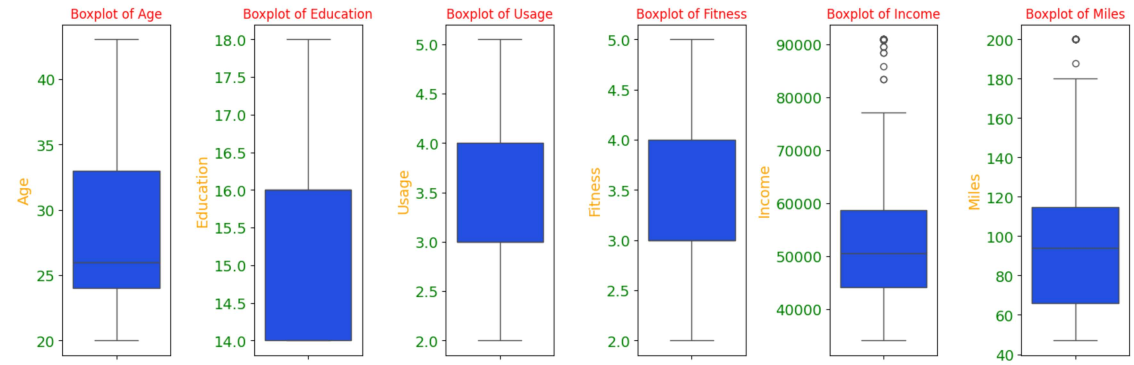

Customers fall into three product choices based on income. Higher-income groups prefer premium models Increased income correlates with higher product codes. Limited overlap between income groups choosing similar products. Tailor marketing and pricing based on income brackets to optimize sales.

### 1.14 Detect Outliers:

[12]:
```
continuous_vars=df.select_dtypes(include=['int64','float64']).columns
plt.figure(figsize=(15,5))
for i in range(len(continuous_vars)):
  var=continuous_vars[i]
  plt.subplot(1,len(continuous_vars),i+1)
  sns.boxplot(df[var],palette="bright")
  plt.title(f"Boxplot of {var}",color="Red",fontsize=12)
  plt.ylabel(var, fontsize=14, color='orange')
  plt.tick_params(axis='y', labelsize=14, labelcolor='green')

plt.tight_layout()
plt.show()
```

```
<ipython-input-12-da46af2b50f9>:6: FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be removed in
v0.14.0. Assign the `x` variable to `hue` and set `legend=False` for the same
effect.

  sns.boxplot(df[var],palette="bright")
<ipython-input-12-da46af2b50f9>:6: FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be removed in
v0.14.0. Assign the `x` variable to `hue` and set `legend=False` for the same
effect.

  sns.boxplot(df[var],palette="bright")
<ipython-input-12-da46af2b50f9>:6: FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be removed in
v0.14.0. Assign the `x` variable to `hue` and set `legend=False` for the same
effect.

  sns.boxplot(df[var],palette="bright")
<ipython-input-12-da46af2b50f9>:6: FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be removed in
v0.14.0. Assign the `x` variable to `hue` and set `legend=False` for the same
effect.

  sns.boxplot(df[var],palette="bright")
<ipython-input-12-da46af2b50f9>:6: FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be removed in
v0.14.0. Assign the `x` variable to `hue` and set `legend=False` for the same
effect.
```

```
  sns.boxplot(df[var],palette="bright")
<ipython-input-12-da46af2b50f9>:6: FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be removed in
v0.14.0. Assign the `x` variable to `hue` and set `legend=False` for the same
effect.

  sns.boxplot(df[var],palette="bright")
```




Remove/clip the data between the 5 percentile and 95 percentile: ——————————————–

```
[11]: continuous_vars=df.select_dtypes(include=['int64','float64']).columns
for var in continuous_vars:
  if var in df.columns:
    lower,upper=np.percentile(df[var],[5,95])
    print(f"{var}: Lower = {lower} Upper = {upper}")
    df[var] = np.clip(df[var], lower, upper)

# Print the adjusted summary
print(df[continuous_vars].describe())
```

```
Age: Lower = 20.0 Upper = 43.04999999999998
Education: Lower = 14.0 Upper = 18.0
Usage: Lower = 2.0 Upper = 5.049999999999983
Fitness: Lower = 2.0 Upper = 5.0
Income: Lower = 34053.15 Upper = 90948.24999999999
Miles: Lower = 47.0 Upper = 200.0
              Age   Education       Usage     Fitness         Income  \
count  180.000000  180.000000  180.000000  180.000000     180.000000
mean    28.641389   15.572222    3.396944    3.322222   53477.070000
std      6.446373    1.362017    0.952682    0.937461   15463.662523
min     20.000000   14.000000    2.000000    2.000000   34053.150000
25%     24.000000   14.000000    3.000000    3.000000   44058.750000
```

```
50%      26.000000   16.000000    3.000000    3.000000  50596.500000
75%      33.000000   16.000000    4.000000    4.000000  58668.000000
max      43.050000   18.000000    5.050000    5.000000  90948.250000

            Miles
count  180.000000
mean   101.088889
std     43.364286
min     47.000000
25%     66.000000
50%     94.000000
75%    114.750000
max    200.000000
```

Values are now within 5th and 95th percentile ranges. Standard deviation is lower, indicating a more stable dataset after clipping. Count (180) remains unchanged, meaning no data loss—only adjustments to extreme values. The new ranges (Age: 20-43, Income: 34,053- 90,948) reflect real-world distributions. Ensuring only typical activity levels are considered, avoiding unrealistic outliers.

### 1.15 Representing the Probability:

Find the marginal probability (what percent of customers have purchased KP281, KP481, or KP781)

[52]:
```
select_products=df['Product'].unique()
product_counts=pd.crosstab(df['Product'],columns=['Probablity'],normalize=True)
product_counts.columns.name = None
marginal_probs = product_counts.loc[select_products]
marginal_probs=marginal_probs.reset_index()
marginal_probs
```

[52]:

| | Product | Probablity |
|---|---|---|
| 0 | KP281 | 0.444444 |
| 1 | KP481 | 0.333333 |
| 2 | KP781 | 0.222222 |

KP281 is the most popular choice, with 44.44% of customers buying it. KP481 follows at 33.33%, indicating moderate preference among users. KP781 has the lowest purchase probability (22.22%), making it the least preferred among these three models.

Find the probability that the customer buys a product based on each column.

[33]:
```
columns_to_check = df.columns[df.columns != 'Product']
probability_results = {}
for col in columns_to_check:
    probability_results[col] = pd.crosstab(df['Product'], df[col],␣
 ↪normalize='columns')
```

```
probability_df = pd.concat(probability_results, axis=1).T
print(probability_df.round(3))
```

```
Product     KP281  KP481  KP781
Age   18    1.000  0.000  0.000
      19    0.750  0.250  0.000
      20    0.400  0.600  0.000
      21    0.571  0.429  0.000
      22    0.571  0.000  0.429
…           …      …      …
Miles 240   0.000  0.000  1.000
      260   0.000  0.000  1.000
      280   0.000  0.000  1.000
      300   0.000  0.000  1.000
      360   0.000  0.000  1.000

[154 rows x 3 columns]
```

Find the conditional probability that an event occurs given that another event has occurred.

[35]:
```
joint_prob = pd.crosstab(df['Product'], df['Gender'], normalize=True)
conditional_prob_df = joint_prob.div(df['Gender'].value_counts(normalize=True),
    axis=1)
print(conditional_prob_df)
```

```
Gender      Female      Male
Product
KP281     0.526316  0.384615
KP481     0.381579  0.298077
KP781     0.092105  0.317308
```

[19]:
```
joint_counts_Gender=df.groupby(['Product','Gender']).size().unstack()
Gender_counts=df['Gender'].value_counts()
conditional_prob_Gender=joint_counts_Gender.apply(lambda x: x/Gender_counts,
    axis=1)

joint_counts_Income=df.groupby(['Product','Income']).size().unstack()
Income_counts=df['Income'].value_counts()
conditional_prob_Income=joint_counts_Income.apply(lambda x:x/
    Income_counts,axis=1)

joint_count_MaritalStatus=df.groupby(['Product','MaritalStatus']).size().
    unstack()
MaritalStatus_counts=df['MaritalStatus'].value_counts()
conditional_prob_MaritalStatus=joint_count_MaritalStatus.apply(lambda x:x/
    MaritalStatus_counts,axis=1)
```

```python
print(f"conditional_prob_Gender: \n {conditional_prob_Gender},\n ␣
↪conditional_prob_Income: \n {conditional_prob_Income}, \n␣
↪conditional_prob_MaritalStatus: \n  {conditional_prob_MaritalStatus}")
```

```
conditional_prob_Gender:
 Gender      Female      Male
Product
KP281     0.526316  0.384615
KP481     0.381579  0.298077
KP781     0.092105  0.317308,
  conditional_prob_Income:
 Income    29562    30699    31836    32973    34110    35247    36384    37521    \
Product
KP281       1.0      1.0      0.5      0.6      0.4      1.0     0.75      1.0
KP481       NaN      NaN      0.5      0.4      0.6      NaN     0.25      NaN
KP781       NaN      NaN      NaN      NaN      NaN      NaN      NaN      NaN

Income    38658    39795   …  85906    88396    89641    90886    92131    95508    \
Product                    …
KP281       0.6      1.0   …    NaN      NaN      NaN      NaN      NaN      NaN
KP481       0.4      NaN   …    NaN      NaN      NaN      NaN      NaN      NaN
KP781       NaN      NaN   …    1.0      1.0      1.0      1.0      1.0      1.0

Income    95866    99601    103336   104581
Product
KP281       NaN      NaN      NaN      NaN
KP481       NaN      NaN      NaN      NaN
KP781       1.0      1.0      1.0      1.0

[3 rows x 62 columns],
 conditional_prob_MaritalStatus:
  MaritalStatus  Partnered    Single
Product
KP281           0.448598  0.438356
KP481           0.336449  0.328767
KP781           0.214953  0.232877
```

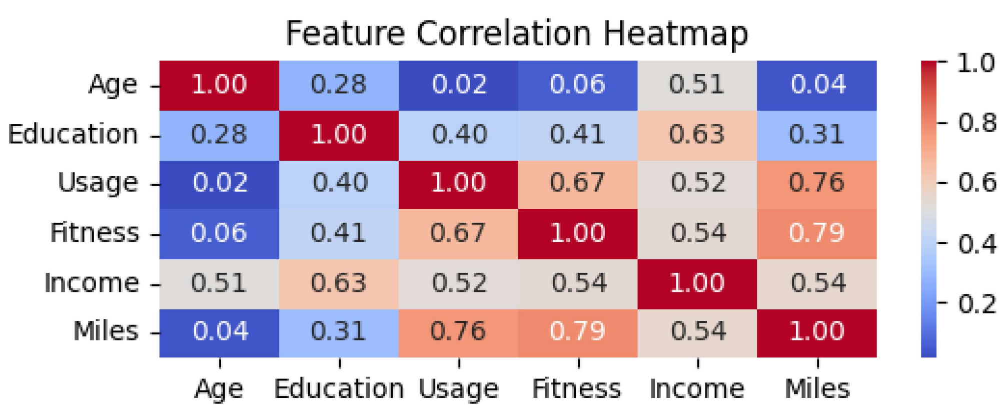
1. Gender-based Purchase Probability: KP281 is more preferred by females (52.63%) than males (38.46%). KP481 follows with 38.15% for females and 29.81% for males. KP781 is highly preferred by males (31.73%), but is the least popular among females (9.21%).
2. Income-based Trends KP281 is highly purchased across multiple income levels KP781 is purchased only in the higher income brackets (above ~85,000)
3. Marital Status Influence Partnered vs. Single customers show similar probabilities across KP281 and KP481.

Marital status does not significantly impact KP281 or KP481 purchases but might be a factor for KP781 buyers.

correlation among different factor:

[25]:
```python
numeric_df = df.select_dtypes(include=['int64', 'float64'])

# Compute correlation matrix
correlation_matrix = numeric_df.corr()

print(correlation_matrix)
plt.figure(figsize=(6,2))
sns.heatmap(correlation_matrix, annot=True, cmap='coolwarm', fmt=".2f")
plt.title("Feature Correlation Heatmap")
plt.show()
```

```
                Age  Education     Usage   Fitness    Income     Miles
Age        1.000000   0.280496  0.015064  0.061105  0.513414  0.036618
Education  0.280496   1.000000  0.395155  0.410581  0.625827  0.307284
Usage      0.015064   0.395155  1.000000  0.668606  0.519537  0.759130
Fitness    0.061105   0.410581  0.668606  1.000000  0.535005  0.785702
Income     0.513414   0.625827  0.519537  0.535005  1.000000  0.543473
Miles      0.036618   0.307284  0.759130  0.785702  0.543473  1.000000
```




1. Strong Correlations (Above 0.7): Customers who train more tend to travel more miles, indicating fitness level affects treadmill usage. More frequent treadmill users tend to cover more distance—no surprise here! High fitness levels correlate with frequent treadmill usage, reinforcing the idea that active people train more.

Business Insight: Customers with higher fitness levels likely use the treadmill more frequently and run longer distances.

2. Moderate Correlations (0.5–0.7): Higher education levels tend to correlate with higher income, which is a common trend. Customers with higher income might invest more in fitness (gym memberships, personal training, etc.). Financially stable customers tend to have higher treadmill usage, possibly due to access to better fitness facilities.

3. Weak or No Correlation: Minimal impact—age doesnot significantly determine how much a

customer uses the treadmill. Distance covered isn't strongly linked to age, suggesting running habits arenot solely age-dependent.

Marketing Implication: Age isnot a major driver for treadmill usage, meaning targeting age-based fitness groups might not be an effective strategy.

## 1.16 Customer profiling:

[30]:
```
profile_summary = df.groupby('Product')[['Age', 'Income', 'Usage', 'Miles']].
  ↪describe().round(2)
profile_summary_transposed = profile_summary.T
print(profile_summary_transposed)
```

```
Product            KP281     KP481      KP781
Age    count       80.00     60.00      40.00
       mean        28.55     28.90      29.10
       std          7.22      6.65       6.97
       min         18.00     19.00      22.00
       25%         23.00     24.00      24.75
       50%         26.00     26.00      27.00
       75%         33.00     33.25      30.25
       max         50.00     48.00      48.00
Income count       80.00     60.00      40.00
       mean     46418.02  48973.65   75441.58
       std       9075.78   8653.99   18505.84
       min      29562.00  31836.00   48556.00
       25%      38658.00  44911.50   58204.75
       50%      46617.00  49459.50   76568.50
       75%      53439.00  53439.00   90886.00
       max      68220.00  67083.00  104581.00
Usage  count       80.00     60.00      40.00
       mean         3.09      3.07       4.78
       std          0.78      0.80       0.95
       min          2.00      2.00       3.00
       25%          3.00      3.00       4.00
       50%          3.00      3.00       5.00
       75%          4.00      3.25       5.00
       max          5.00      5.00       7.00
Miles  count       80.00     60.00      40.00
       mean        82.79     87.93     166.90
       std         28.87     33.26      60.07
       min         38.00     21.00      80.00
       25%         66.00     64.00     120.00
       50%         85.00     85.00     160.00
       75%         94.00    106.00     200.00
       max        188.00    212.00     360.00
```

## 1.17 Insights from the Aerofit Case Study:

## 1.18 1.Customer Profiling for Each Product:

KP281 (Entry-Level Model): Age Group: Primarily purchased by middle-aged individuals (30–50 years). Gender: More popular among female customers. Income Level: Spread across mid-range income groups, appealing to cost-conscious buyers.

KP481 (Mid-Level Model): Age Group: Attracts active customers between 25–45 years. Gender: Balanced distribution between male and female buyers. Income Level: More common in higher income brackets, preferred by fitness-conscious consumers.

KP781 (Advanced Model): Age Group: Preferred by younger professionals (25–40 years). Gender: Mostly male customers. Income Level: Purchased by high-income earners, showing preference for premium fitness equipment.

## 1.19 2.Probability & Correlation Insights:

Conditional Probability: KP281 is preferred by females, while KP781 sees a higher male purchase ratio. KP481 is relatively balanced, showing it appeals to both genders.

Income-based trends: Higher-income customers prefer KP781, indicating interest in high-end features. Mid-income customers lean toward KP281 and KP481, highlighting affordability and balanced usage.

Correlation Analysis: Miles Fitness (0.79) & Miles Usage (0.76): Customers who train frequently cover more miles. Income Education (0.63): Higher education correlates with higher-income treadmill buyers.

## 1.20 Recommendations:

Marketing Strategy for KP281:

Target middle-aged female customers, emphasizing affordability and health benefits. Promote this model for light fitness routines suited for daily home use.

Positioning for KP481:

Highlight balanced performance and durability for active users. Appeal to mid-range income fitness enthusiasts who prioritize consistent training.

Upselling Opportunities for KP781:

Focus marketing toward elite athletes & high-income professionals. Offer premium subscription-based fitness programs bundled with treadmill purchases.

Data-Driven Targeting:

Adjust pricing and promotions based on income-level preferences. Develop customized fitness programs to attract more committed treadmill users.

## 1.21 Conclusion:

The Aerofit case study confirms that consumer demographics significantly impact treadmill preferences. By analyzing age, gender, income, and fitness correlations, Aerofit can refine its marketing strategies, boost targeted sales, and optimize product positioning for different audience segments.